IN THE CLAIMS

Please amend the claims as follows:

1. (original) A method of modelling a state machine comprising a first state model (client1), and a second state model (mf1) implementing a function call, the method comprising, in response to an event in the first state model instructing the firing of the function call, implanting the second state model in the first state model.

2. (original) A method according to claim 1, in which the second state model is absent of history information.

3. (currently amended) A method according to ~~either preceding claim~~claim 1, in which the second state model contains one or more clusters (client1, client2).

4. (currently amended) A method according to ~~any preceding claim~~claim 1, in which the second state model contains one or more sets (system).

5. (currently amended) A method as claimed in ~~any preceding claim~~claim 1, in which the second state model contains two or more leafstates (f1_a, f1_b) having one or more event driven transitions (β) therebetween.

6. (original) A method as claimed in claim 5, in which one or more of the transitions fires a notification event (pending_f, final_notif_g).

7. (currently amended) A method as claimed in ~~any preceding claim~~claim 1, in which the second state model is implanted over an explicit marker state (calling) of the first state model.

8. (currently amended) A method as claimed in ~~any of claims 1 to 6~~claim 1, in which the second state model is implanted over an implicit marker state of the first state model.

9. (currently amended) A method as claimed in claim 7~~ or claim 8~~, in which the second state model is deleted on completion.

10. (original) A method as claimed in claim 9, in which the model allows the entering of a state in the first model local to the

caller of the second state model only on deletion of the second state model.

11. (currently amended) A method as claimed in ~~any of claims 7 to 10~~claim 7, in which local declarations and/or scoping operators are used in the second state model.

12. (original) A method as claimed in claim 11, in which a return event from the second state model uses a "back" scoping operator ($).

13. (currently amended) A method as claimed in ~~any of claims 1 to 5~~claim 1, in which the second state model is implanted in free-space.

14. (original) A method as claimed in claim 13, in which the lifetime of the second state model is independent of any other model.

15. (currently amended) A method as claimed in claim 13~~ or claim 14~~, in which the second state model is implanted local to the caller of the second state model.

16. (currently amended) A method as claimed in ~~any of claims 13 to 15~~claim 13, in which notification events from the second state model are global.

17. (currently amended) A method as claimed in ~~any of claims 13 to 17~~claim 13, in which the second state model is deleted on transition to a terminator forming part thereof.

18. (currently amended) A method as claimed in ~~any preceding claims~~claim 1, in which events occurring in the second state model are parameterised.

19. (currently amended) A method as claimed in ~~any preceding claim~~claim 1, in which events occurring in the first state model are parameterised.

20. (currently amended) A computer program containing instructions for a computer to carry out the method of ~~any of claims 1 to 18~~claim 1.

21. (original) A computer program as claimed in claim 19, further comprising instructions for a computer to generate an executable program exhibiting the same behaviour as the state model.

22. (currently amended) A computer program as claimed in claim 19 ~~or claim 20~~, further comprising instructions for a computer to generate tests with an oracle, for testing an implementation conformant to the behaviour of the state model.

23. (original) A computer programmed with the computer program of claim 19.

24. (original) Apparatus for modelling a state machine comprising a first state model (client1) and a second state model (mf1) implementing a function call, the apparatus comprising means responsive to an event in the first state model instructing the firing of the function call, for implanting the second state model in the first state model.

25. (original) Apparatus according to claim 24, in which the second state model contains one or more clusters (client1, client2).

26. (currently amended) Apparatus according to claim 24~~ or claim 25~~, in which the second state model contains one or more sets (system).

27. (currently amended) Apparatus according to ~~any of claims 24 to 26~~claim 24, in which the second state model contains two or more leafstates (f1_a, f1_b) having one or more event driven transitions (β) therebetween.

28. (currently amended) Apparatus as claimed in ~~any of claims 24 to 27~~claim 24, in which one or more of the transitions fires a notification event (pending_f, final_notif_g).

29. (currently amended) Apparatus as claimed in ~~any of claims 24 to 28~~claim 24, in which the second state model is implanted over an explicit marker state (calling) of the first state model.

30. (currently amended) Apparatus as claimed in ~~any of claims 24 to 28~~claim 24, in which the second state model is implanted over an implicit marker state of the first state model.

31. (currently amended) Apparatus as claimed in claim 29~~ or claim 30~~, comprising means for deleting the second state model on completion.

32. (original) Apparatus as claimed in claim 31, in which the model allows the entering of a state in the first model local to the caller of the second state model only on deletion of the second state model.

33. (currently amended) Apparatus as claimed in ~~any of claims 24 to 28~~claim 24, in which the second state model is implanted in free-space.

34. (original) Apparatus as claimed in claim 33, in which the second state model is implanted local to the caller of the second state model.

35. (currently amended) Apparatus as claimed in claim 33~~ or claim 34~~, comprising means for deleting the second state model on transition to a terminator forming part thereof.